AIAA SciTech
5-9 January 2015, Kissimmee, Florida
AIAA Infotech @ Aerospace

RUMS - Realtime Visualization and Evaluation of Live, Virtual, Constructive Simulation Data

George Soler* and Srba Jovic†
SAIC/NASA Ames Research Center, Moffett Field, CA, 94035

and

James R. Murphy‡
NASA Ames Research Center, Moffett Field, CA, 94035

Live, Virtual, Constructive (LVC) environments are designed to integrate flight assets with simulated aircraft and airspace to support human in the loop simulation and flight test activities. NASA's Unmanned Aircraft System (UAS) Integration into the National Airspace System (NAS) Project is developing an LVC environment to research the display, communication, and acceptability of detect and avoid concepts to the pilot. The simulations and flight tests are conducted over a nationwide network with components communicating via software gateways that translate the messages and provide a location to access and collect data. The data collected is not only archived for post-processing and analysis, but also processed in real-time by the Remote User Monitor Service (RUMS), which allows real time viewing of the information via a web browser. This real-time dynamic viewing of the data allows test conductors and researchers to monitor the health of the distributed system and the data being collected not only in situ where tests are conducted but also at remote locations. This paper discusses the design approach and general software architecture of the RUMS application.

Nomenclature

ADRS	=	Aeronautical Data Link and Radar Simulator
DSRL	=	Distributed Simulation Research Laboratory
HLA	=	High Level Architecture (simulation middleware)
LVC	=	Live, Virtual, Constructive describing the simulation environment
MACS	=	Multi-Aircraft Control System (a pilot and air traffic emulator)
NAS	=	National Airspace System
RUMS	=	Remote User Monitoring System
UAS	=	Unmanned Aircraft System
XML	=	Extensible Markup Language (programming language)

I. Introduction

THERE is an increasing desire for a capability to fly government and commercial Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS).[1] Under the UAS Integration into the NAS project, NASA is conducting simulations and flight tests to determine how to reduce barriers that limit the routine access of UAS into the NAS. Due to the complexity of these events, NASA has built a distributed Live, Virtual, Constructive (LVC)

* Software Developer, SAIC/NASA Ames Research Center MS 243-6, 94035
† Software Manager, SAIC/NASA Ames Research Center MS 243-6, 94035
‡ Project Engineer, NASA Ames Research Center MS 243-1, AIAA Associate Fellow

environment that supports mixing of live aircraft into simulated air traffic control airspace[2]. LVC constructs have been used to integrate live and virtual assets for training and efficient flight-testing for many years throughout the Department of Defense and have been shown to facilitate simulation among distributed partners.[3,4,5] The NASA LVC system is comprised of flight components, facilities, and software assets integrated across multiple NASA Centers. The management of a simulation spread over separate facilities is a challenge not only in the conduct of a simulation or flight-test, but also with the monitoring of the health of the system, the collection of the research data, and real time monitoring of research parameters. In order to protect the integrity of the components connected to the overall LVC environment, the LVC network must maintain a level of security that is agreed upon by each of the participating facilities. In practice, the LVC network is maintained at a higher security level then a typical research environment, which adds to the challenge of data distribution.

A dispersed group of simulation end-users and stakeholders in a distributed simulation environment could benefit from a tool that monitors the simulation's progress at run time without the need to be connected directly to the simulation environment. The UAS in the NAS project has designed and integrated a data collection and monitoring capability into the LVC distributed test environment that provides the means to remotely access and evaluate the system in real-time. This capability has two primary uses: 1.) to ensure the validity of data being collected by providing the test team a mechanism for monitoring of the health of the test system in near real-time, and 2.) to distribute the data to users not on the test network by providing access to the data from a database server.

The Remote User Monitoring System (RUMS) capability was developed at the Distributed Simulation Research Laboratory (DSRL), Aerospace Simulation Research and Development Branch (SimLabs), NASA Ames Research Center to provide a standardized, user-friendly, near real time simulation-monitoring tool concurrently available to all authenticated end-users outside of the test network. RUMS is built using modern web framework technologies that enable the simulation to be monitored in real-time and allow key data to be visualized via a web browser using information being transmitted through the LVC system. It also provides a location to capture the data for post processing, analysis, and redundant backup.

II. Background

A. UAS in the NAS Project

The application of unmanned aircraft to perform national security, defense, scientific, and emergency management functions is driving the critical need for less restrictive access by UAS to the NAS. UAS represent a new capability that will provide a variety of services in the government (public) and commercial (civil) aviation sectors. The growth of this potential industry has not yet been realized due in part to the lack of regulations governing the safe operation of unmanned vehicles in what is now manned airspace.

NASA's UAS Integration into the NAS Project is conducting research that contributes capabilities intended to reduce technical barriers related to the safety of and operational challenges associated with enabling routine UAS access to the NAS.[2] To accomplish this task, NASA will conduct a series of integrated human-in-the-loop and flight test activities that integrate key concepts, technologies and/or procedures in a relevant air traffic environment. The research is guided by the development of the Minimum Operating Performance Standards being drafted by RTCA special committee 228 covering detect and avoid, communication, and display of data to a UAS pilot. Each of these integrated events will build on the technical achievements, fidelity, and complexity of the previous tests and technical simulations, resulting in research data that supports the development of regulations governing the access of UAS into the NAS.

B. Distributed Test Environment

To support these planned simulations and flight tests, NASA developed a distributed LVC test environment. The use of "live", "virtual", or "constructive" to describe simulation components is not well defined since there is no clear division between these categories. A "constructive" component generally has no human interaction with the simulated environment. Instead, scenarios unfold using rule-based decisions that control the interactions between simulated actors. "Virtual" components involve human participants operating simulated systems (e.g., a pilot in a flight simulator). A "live" component involves human participants operating real systems.[3] Figure 1 shows a high level depiction of the LVC Concept of Operation (ConOps) for the UAS in the NAS project simulations and flight tests. The ConOps includes simulation and flight assets from each of the four NASA Aeronautic Centers: Armstrong Flight Research Center, Ames Research Center, Glenn Research Center, and Langley Research Center. The facilities

support flights of manned and unmanned aircraft, flight simulators, air traffic control workstations and ground control display testing.

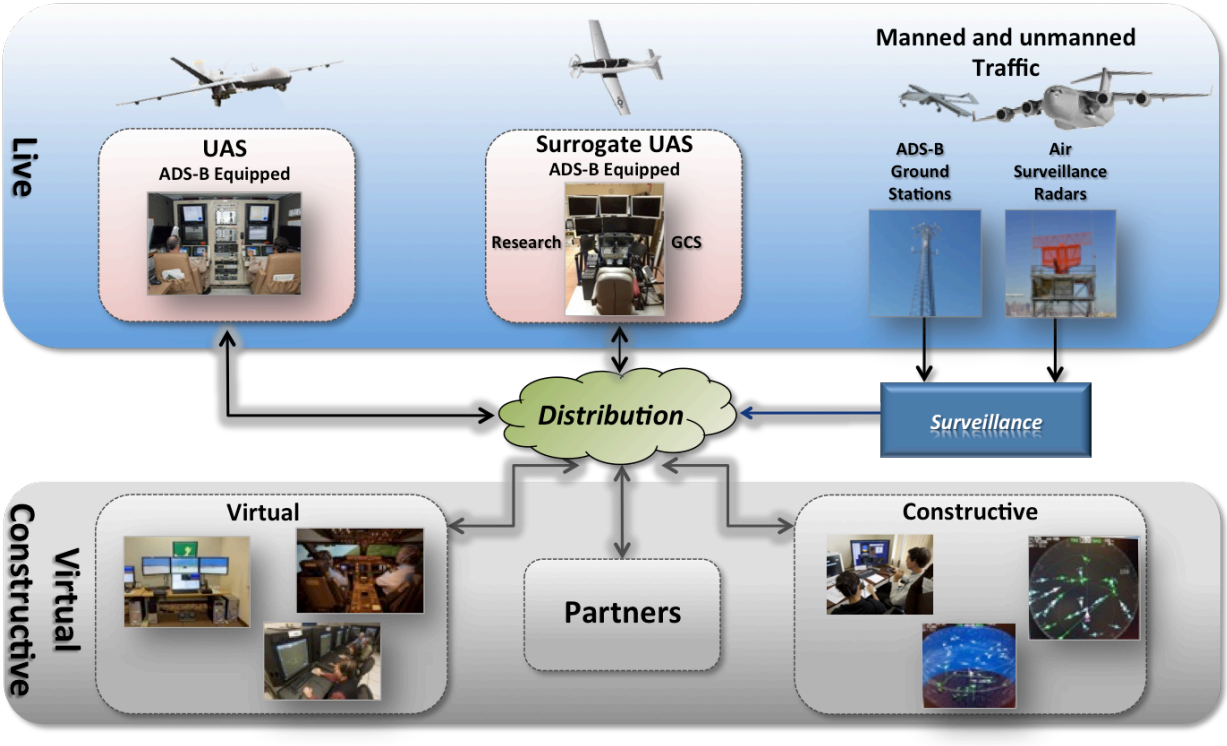

Figure 1. LVC System ConOps. *The LVC integrates distributed assets for a more efficient and relevant test environment*

C. HLA and LVC Gateway

The distribution cloud shown in Figure 1 and the relationship of the RUMS software system is best illustrated through high-level use case description. Figure 2 depicts the connectivity between the primary software components for an LVC system connecting four NASA Centers. In this example, NASA Ames provides virtual air traffic control and aircraft traffic, a live UAS aircraft is provided by NASA Armstrong, a virtual UAS simulator connects from NASA Langley, and a live aircraft is flown from NASA Glenn.

Table 1 provides a short description of the processes used to connect clients to the LVC infrastructure. All participating simulation and flight components are integrated either through a High Level Architecture (HLA) Toolbox interface or an LVC Gateway, which comprise the core LVC infrastructure. HLA is an industry standard that provides a well-defined set of message formats and an application programming interface for conducting air traffic simulations supporting the distribution of the test components among the separate test facilities.[6] For the UAS testing, HLA

Table 1. List of processes used in connecting air traffic data among NASA Centers

Process	Process Description
HLA	General purpose architecture for conducting distributed simulations
LVC Gateway	Message passing process that routes local messages among clients and connects clients to the HLA
Toolbox	Translates messages from the client format to the HLA format
MACS	Provides air traffic workstations and aircraft target generation
MACS Gateway	Message routing program for MACS
RUMS Gateway	Translates messages from the LVC Gateway into a format for the database
Virtual Sandbox	Data storage for the RUMS Server
RUMS Server	External access to LVC Test data via web interface

is managed out of the DSRL at NASA Ames. The HLA used for this Project utilizes a version of the IEEE 1516 standard Pitch portable Real Time Infrastructure High Level Architecture and Federation Object Model middleware.[7] The existing HLA distributed environment has the capability to integrate constructive air traffic that is generated by the Multi-Aircraft Control System (MACS), which also contains a robust air traffic control and pseudo pilot capability.[8] MACS publishes and subscribes to a well-defined set of messages (Flight Plan, Flight State, and Trajectory Intent) through an HLA Toolbox via its own software gateway.

The LVC Gateway was developed to generalize the connection of LVC client programs that do not have an existing HLA Toolbox of their own. It also provides a mechanism for multiple local clients to connect to each other without the need to send messages through the HLA (which may be at a distributed facility).[9] The LVC Gateway was used to connect to a virtual UAS ground control station at NASA Langley, a live UAS (the Ikhana MQ-9) at NASA Armstrong, and a live manned aircraft at NASA Glenn flown as a surrogate UAS aircraft.

The RUMS Server System, which will be described in the next section, operates from the DSRL at NASA Ames. Each LVC Gateway process being monitored requires a separate set of RUMS processes. This is due to the fact that all data messages handled by the LVC Gateway are stored in a separate database. Users of RUMS can select the LVC Gateway to monitor then access the data through the Internet via a common web browser. Figure 2 shows a single RUMS system connected to the virtual ground control station running at NASA Langley.

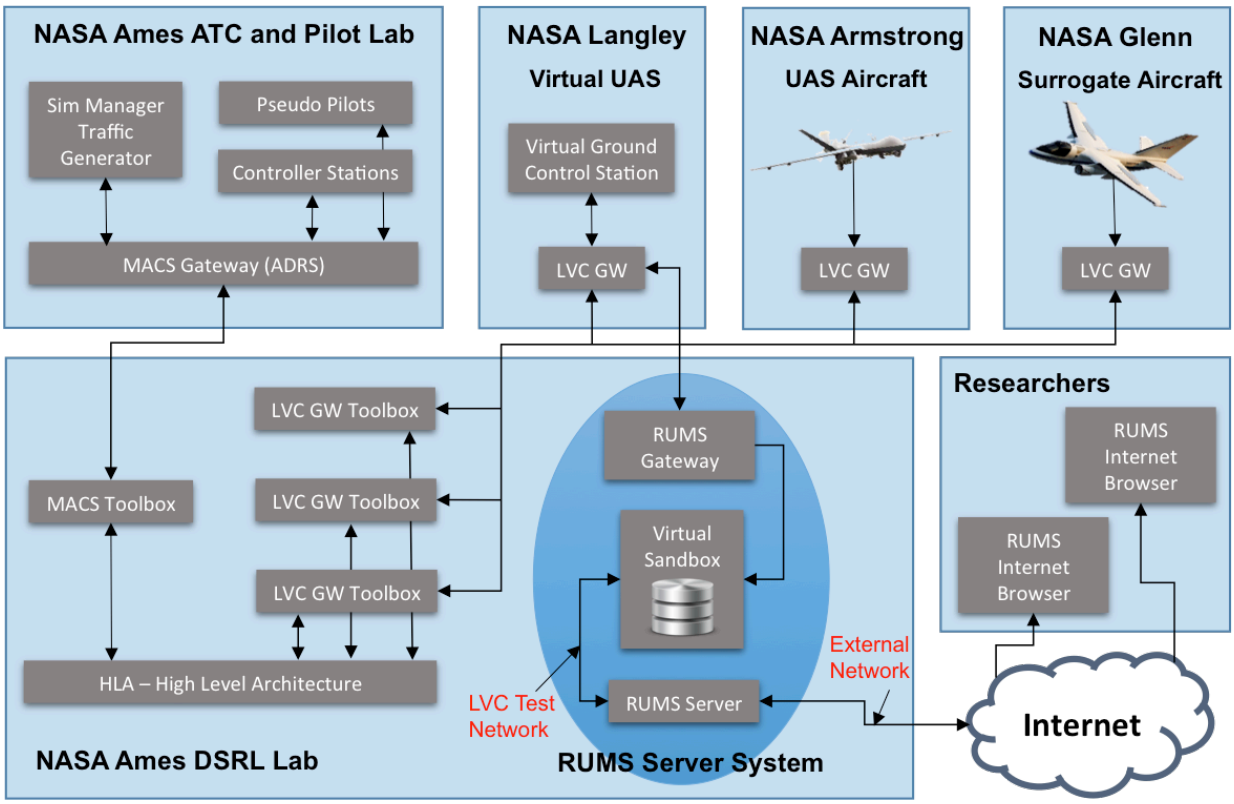

Figure 2. Distributed LVC Architecture. *The LVC system was developed and configured to connect four NASA Centers.*

III. Remote User Monitoring System

Making simulation data available in real-time to all stakeholders in a distributed environment can be challenging. Presenting real-time simulation data in a simple and universally accessible way usually involves a variety of software tools. Disparate monitoring tools may have features that cannot be easily ported between systems. Some tools may be proprietary or may require some form of maintenance by end-users. All this works against uniformity in monitoring.

RUMS is a component of a distributed subsystem within the DSRL Lab. The subsystem depicted in the oval in Figure 2 contains the software components that make up RUMS. The subsystem includes a RUMS Gateway component that provides two-way communications with the LVC, a RUMS Server component that runs a web server, and a component that manages a 'Virtual Sandbox'. In this context a Virtual Sandbox is an abstraction for a pool of objects that represent all the aircraft (live and virtual) that participate in a simulation at any given moment. In essence, it is the local source for all data accessible by RUMS. The machine that runs the RUMS server process has a dual network which bridges between the test data on the LVC test network and the outside world. This allows external users to access to test data while keeping the test network isolated from the external inputs. At this time in the RUMS development process, access control is managed through the discrete data displays developed for research use. Research project data is typically widely disseminated, so controlled data access by user is not a primary concern.

A. RUMS Data Flow

The RUMS Gateway component of the RUMS subsystem receives messages from the LVC Gateway via a socket interface. See Figure 3 for a diagram of the RUMS data flow. The messages contain aircraft state, aircraft flight plans, trajectories, and ancillary messages, such as separation assurance algorithm data. The messages are unpacked according to the protocol specified in the LVC Gateway Message Interface Control Document. When a message is unpacked a new Java object (MpiAcStateMsg) is constructed from the message contents. The RemoteMpiSession class uses Java's notification event model to notify the Gateway's communications hub Observer (GatewayCommHub). The GatewayCommHub class is a client of the Virtual Sandbox component. Once the Gateway communications hub Observer is notified it uses a transactional memory model to update the Virtual Sandbox map of aircraft objects (acMap) with the specific information contained in the message.

The RUMS server component also acts as a client of the Virtual Sandbox. The RUMS client polls the Virtual Sandbox every second to provide the RUMS embedded web server with the data needed to update the displays on a web page. Any web clients that connect to the RUMS server obtain a representation of the data via the various displays in the RUMS browser user interface.

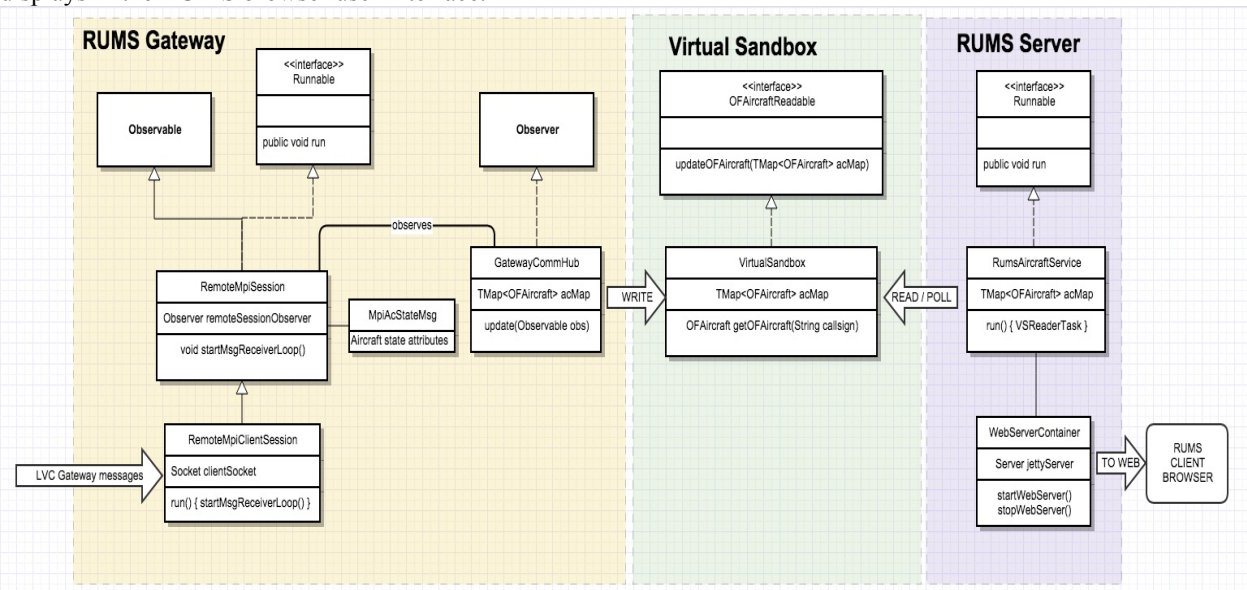

Figure 3. RUMS server UML class diagram. *The RUMS server system is comprised of three separate programs that receive data from the source LVC Gateway, store data for easy access, and provide data to the any clients.*

B. Development

RUMS was designed to provide non-mission critical airspace situation awareness for monitoring air traffic control target information, and to provide a simple set of analysis and diagnostics tools for researchers. The RUMS subsystem was specifically developed to monitor a distributed air traffic simulation where aircraft state data originates independently in real-time from various networked subsystems (In this context, real-time data with one-

second granularity that is not subject to timing deadlines or hardware interrupts). Although RUMS was developed specifically to assist UAS-NAS integration research, the design, framework selection, and the general architecture can be replicated to fit other problem domains.

All the components in the RUMS subsystem are Java applications. RUMS is essentially a web application with a Jetty embedded web server at its core.[10] RUMS is hosted on a dual-home network system to allow incoming client connections from outside the DSRL environment while protecting the DSRL intranet. As with other components in the RUMS subsystem, the behavior of RUMS is controlled by an Extensible Markup Language (XML) configuration properties file. Each component in the subsystem can be reconfigured, enabling or disabling specific features between simulation runs by changing individual properties. This avoids the need to recompile between runs.

Design of the RUMS communications is predicated on the component-based architecture of the RUMS subsystem. RUMS communicates asynchronously with the other components in the subsystem using a two-tier strategy. Tier-one communications consists of targeted messages between components. These involve low volume network traffic of high importance: sporadic topical command and control messages. Throughput requirements are low but message integrity and persistence is critical and message queuing is needed. Tier-one messaging is handled by a Java Messaging Service[11] message-oriented middleware server that runs on a dedicated host. Tier-two communications are based on the transactional memory client-server paradigm. This strategy uses a transactional object model of the aircraft state class to continuously map low-level memory transfers between the RUMS subsystem's networked processes. RUMS uses the ObjectFabric transactional memory client-server framework.[12] The ObjectFabric server is embedded in the Virtual Sandbox component, and it usually runs on a dedicated host. The ObjectFabric server provides the Virtual Sandbox the means to update the aircraft states and distribute a representation of all the aircraft that participate in the simulation. The ObjectFabric transactional memory object model description is stored in XML format. As a client of the ObjectFabric server, RUMS runs a thread that continuously polls the ObjectFabric server and resolves the ObjectFabric memory map into multiple message updates that are propagated to the rest of the RUMS client code.

C. RUMS User Interface and Data Monitoring

For the user interface aspects of the design, RUMS uses the Vaadin web development framework.[13] The Vaadin framework interacts transparently with the Google Web Toolkit.[14] The Vaadin programming interface allows rich user interface feature development in pure Java with minimal incursions into Javascript.[15] RUMS's main user interface features in the web browser include a simulation status page, a traffic display map, a separation display page, and a scenario playback page. Each page has embedded controls and display items, which, taken together, allow for a rich user-friendly simulation data viewing experience. Data monitoring with RUMS is done through the web interface. At present, there are three main tabs within the browser page that cluster monitoring functions. It shows the current DSRL scenario and data run status. It may also show a schedule, contact information, and other general information. The status content changes dynamically, updated automatically to reflect current information.

The traffic display tab shows a plan-view map with aircraft markers superimposed on top (see Figure 4). The markers are updated automatically in near real time and move at approximately one-second intervals. Each marker is associated with a corresponding aircraft. The aircraft's callsign is shown below the aircraft icon and the altitude is shown below the callsign. A climb (up/green) or descent (down/red) arrow is shown next to the altitude. To view detailed information about a particular aircraft, click on the aircraft marker or select its callsign from the pull-down menu. The traffic display map can be panned and zoomed. Currently, there are two map flavors: the default remote monitor service map (stylized with subdued colors), and a standard terrain map.

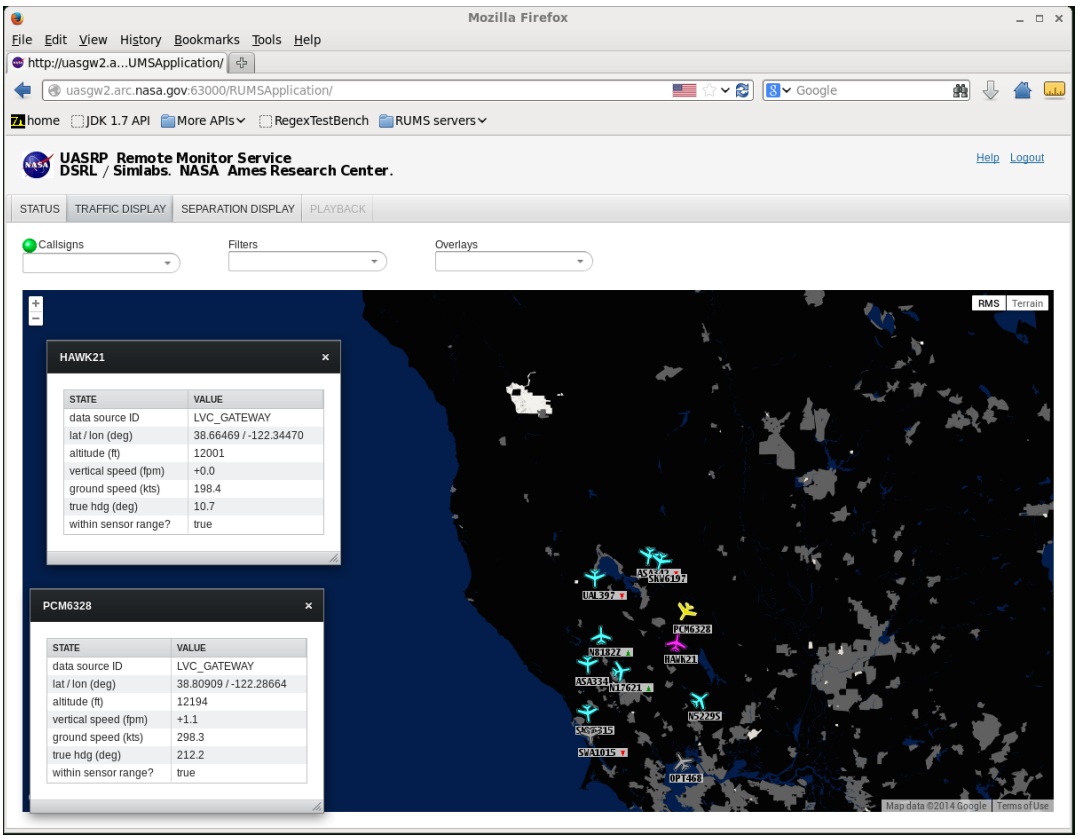

Figure 4. RUMS Webserver Traffic display tab. *The RUMS Web Browser access has a built-in traffic display that shows selected aircraft, aircraft information, and geographic display features in subdued colors.*

The separation display tab shows various information panels related to aircraft range, separation, and proximity alerts. The main panel consists of a radar-style display, which shows the targets from an ownship-centric frame of reference. It also provides a menu option to display the selected alert with a basic plot of the information as shown in Figure 5. Since RUMS has access to not only the state, but also algorithmic derived data being sent among the distributed clients, it can be configured to display other data as requested by researchers.

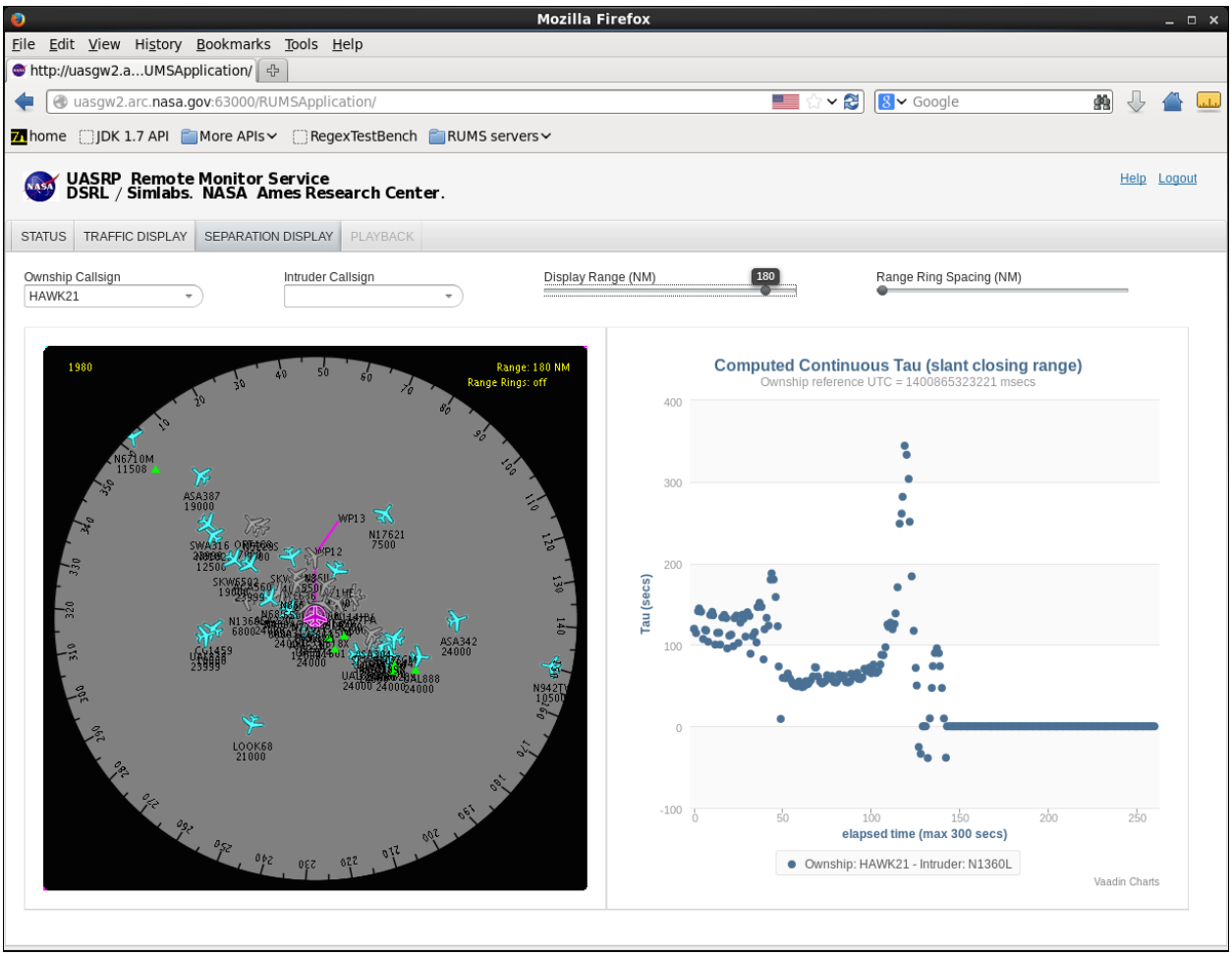

Figure 5. RUMS Webserver Separation display tab. *The RUMS Web Browser access presently has a Separation tab that presents a set of data pertaining to the separation of two selected aircraft.*

IV. RUMS Usage and Next Steps

The RUMS Server and web application are presently used by the UAS Integration in the NAS project to monitor the live data flow and collection of messages sent among the distributed facilities. Since RUMS taps into the same data streams used to pass messages between facilities, it can be used to identify potential data issues and provides an offline near real-time check of the data collected during testing. In addition, since it is accessible by a common web browser via the Internet, it allows the data to be accessed and displayed more easily outside the simulation facilities. Currently, the RUMS displays are being evaluated to support external demonstrations.

One of the more powerful uses of RUMS is a quick view display of internal simulation algorithm data. As shown in the Separation display tab in Figure 5, RUMS has direct access to not only static state data of the aircraft but also derived algorithmic data. Using the remote access to the data through the web browser, developers can quickly generate views of the simulation data for use by researchers and simulation observers. This can be done without modification to the existing simulation displays, alleviating the need to make changes to previously tested software.

Moving forward, the RUMS development team is working with the UAS-NAS project researchers to define a display that can be used by the Test Director of an upcoming flight test to show the relative paths of the live aircraft and their closest point of approach. Other predefined displays are also being discussed with the researchers. To address the scalability of the RUMS system, the number of RUMS web clients the Server can support is being investigated.

Following the implementation of new display ideas from the researchers, a long-term goal of RUMS is to allow for display of the data based on dynamic user selection (i.e. dynamic displays based on user selection of the

available data). This type of dynamic data access has been successful in the past, notably by the Air Force Research Lab and NASA using JView.[16,17] At this time in the RUMS development process, however, access control is managed through the discrete data displays developed for research use. Dynamic data access would require that data access control be scrutinized more closely. Integrating these concepts and technologies may provide RUMS with a basis for the development of truly flexible data display.

References

[1] FAA Modernization and Reform Act of 2012. Feb. 1, 2012. Accessed May 15, 2013 from http://www.gpo.gov/fdsys/pkg/CRPT- 112hrpt381/pdf/CRPT-112hrpt381.pdf

[2] NASA Technology Development Project Plan, Unmanned Aircraft Systems (UAS) Integration in the National Airspace System (NAS), 31 January 2013

[3] DoD: "Modeling and Simulation Master Plan", DoD 5000.59P, Oct 1995

[4] Henninger, Amy E., Cutts, Dannie, Loper, Margaret, etal, "Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report", Institute for Defense Analysis, Sept, 2008

[5] Lutz, R., LeSueur, K., Fast, P., Graeff, R., Simolte, A., Rutledge, J., and Motti, R., "A Persistent LVC Simulation Environment for UAS Airspace Integration," The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), December 2010

[6] Dahmann, J. S., Fujimoto Richard M., Weatherly, Richard M., "The Department of Defense High Level Architecture," Proceedings of the 1997 Winter Simulation Conference, pages 142-149, 1997

[7] Software website: http://www.pitch.se/prti, May 2014

[8] Prevot, T, "Exploring the many perspectives of distributed air traffic management: The Multi Aircraft Control System: MACS", International Conference on Human– Computer Interaction in Aeronautics, HCI–Aero 2002, 23–25.

[9] Murphy, J. Jovic, S., Otto, N., "Message Latency Characterization of a Distributed Live, Virtual, Constructive Simulation Environment," In Press, AIAA Infotech@Aerospace Conference, January 2015

[10] Software website: http:// wiki.eclipse.org/Jetty, August 2014

[11] Software website: http:// docs.oracle.com/javaee/6/tutorial/doc/bncdr.html, August 2014

[12] Software website: http://objectfabric.com, August 2014

[13] Software website: http://www.vaadin.com, August 2014

[14] Software website: http://www.gwtproject.org, August 2014

[15] "What is Javascript?" retrieved from software website: http://media.wiley.com, November 2014

[16] "Advanced Visualization and Interactive Displays (AVID)", AFRL-RI-RS-TR-2009-115, Air Force Research Laboratory, April 2009.

[17] Murphy, J., and Hinton, S., "User Centered, Application Independent Visualization of National Airspace Data," AIAA-2011-1407, AIAA Infotech@Aerospace 2011, March 2011.